IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPELLANT:	Dee Gardiner
SERIAL NO.:	09/694,411
FILED:	October 23, 2000
CONFRM. NO.:	9053
FOR:	METHOD FOR REDUCING TRANSPORT DELAY IN AN IMAGE GENERATOR
ART UNIT:	2628
EXAMINER:	Roberta D. Prendergast
DOCKET NO:	2469-T9180

CERTIFICATE OF DEPOSIT

DATE OF DEPOSIT: August 27, 2007

I hereby certify that this paper or fee (along with any paper or fee referred to as being attached or enclosed) is being electronically deposited using EFS Web with the United States Patent Office on the date indicated above.

/Steve M. Perry/

Steve M. Perry

APPELLANTS' APPEAL BRIEF UNDER 37 C.F.R. § 41.37

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450
Mail Stop Appeal Brief – Patents

Sir:

Appellants submit this Appeal Brief in connection with their appeal from the final rejection of the Patent Office, mailed February 27, 2007, in the above-identified application. A Notice of Appeal was filed on June 26, 2007.

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The real party in interest of this application is Rockwell Collins, Inc., 400 Collins Rd N.E., Cedar Rapids, IA 52498.

II. RELATED APPEALS AND INTERFERENCES

Appellants and Appellants' legal representatives know of no other appeals or interferences that will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III. STATUS OF CLAIMS

Claims 10-14, 24, 26-29, 32 and 34-35 remain pending. Claims 1-9, 15-23, 25, 30, 31, 33, and 36-38 have been previously canceled. Claims 10-14, 24, 26-29, 32 and 34-35 stand rejected. The claims on appeal in this application are 10-14, 24, 26-29, 32 and 34-35.

IV. STATUS OF AMENDMENTS

Claims 10, 24, 32, 34 and 35 were amended after the Office Action mailed February 27, 2007 and were entered to place the application in a better condition for appeal. Claims 25, 30-31, 33 and 36-38 were cancelled without prejudice. The Examiner has withdrawn the 35 USC § 112, second paragraph rejection of claims 25, 30, 31, 33-35 and 37. Claims 10-14, 24, 26-29, 32, 34 and 35 stand rejected under 35 USC 102(e) and 103(a) as written in the Office Action mailed February 27, 2007.

V. SUMMARY OF CLAIMED SUBJECT MATTER

10. (previously presented) A method for enabling a single pixel frame buffer for simultaneous rendering and display in a computer image generator, comprising the steps of:

(a) dividing a geometry buffer into a plurality of screen bins 152 (page 10, lines 13-15);

(b) storing primitives 154 in each screen bin 152 the primitives touch (page 10, lines 15-17);

(c) rendering 144 the screen bins 152 by row from top to bottom, into the single pixel frame buffer (page 11, lines 3-5);

(d) displaying 142 at least one row of screen bins 152 rendered before the rendering of all the screen bins has completed, wherein the displaying of the screen bins 152 takes place after a selected portion of the screen bins for a current field have been rendered. (page 10, lines 5-11).

24. (currently amended) A method for enabling a single pixel frame buffer for simultaneous rendering and display in a computer image generator, comprising the steps of:

(a) dividing a geometry buffer into a plurality of screen bins 152 (page 10, lines 13-15);

(b) storing primitives 154 in each screen bin 152 containing a portion of the primitive (page 10, lines 15-21, FIG. 6);

(c) rendering 144 the screen bins 152, by row from top to bottom, into the single pixel frame buffer (page 11, lines 3-5); and

(d) displaying 142 at least one rendered screen bin 152 when the rendering of the screen bins for the single pixel frame buffer is at least ½ completed (page 14, lines 20-21).

32. (currently amended) An image generator with a single pixel frame buffer enabled for simultaneous rendering and display, comprising:

(a) a geometry buffer divided into a plurality of screen bins 152 (page 10, lines 13-15);

(b) a plurality of primitives 154, stored in all of the screen bins 152 that touch a screen region defined by the screen bin 152 (page 10, lines 15-21);

(c) a rendering engine 144, configured to render the primitives in the screen bins 152 into the single pixel frame buffer by row and from top to bottom (page 11, lines 3-14);

(d) a display processor 142, configured to display at least one rendered screen bin 152 on a display screen 150 before the rendering engine 144 has completed rendering all the screen bins 152 (page 8, lines 11-12; page 11, lines 3-7); and

(e) wherein the display processor 142 begins to display the screen bins 152 rendered when the rendering 144 of the screen bins 152 is at least ½ complete (page 14, lines 20-21).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The issues presented for review are:

a. whether independent claim 10 and dependent claims 11 and 13 are unpatentable under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 5,864,342 (hereinafter "Kajiya");

b. whether dependent claims 12 and 14 are unpatentable under 35 U.S.C. § 103(a) as being obvious over Kajiya in view of U.S. Patent No. 6,853,381 (hereinafter "Grigor").

c. whether independent claims 24 and 32 and dependent claims 26-29, 32, 34 and 35 are unpatentable under 35 U.S.C. § 103(a) as being obvious over Kajiya in view of U.S. Patent No. 6,316,974 (hereinafter "Taraci");

VII. ARGUMENT

A. Prosecution History

The present application was filed as an original utility application on October 23, 2000 under the title METHOD FOR REDUCING TRANSPORT DELAY IN AN IMAGE GENERATOR. Twenty three claims were presented. The application was assigned Serial No. 09/694,411.

A Notice of Allowance was mailed on May 5th, 2003. Claims 1-23 were allowed. The Primary Examiner, Cliff Vo, stated that reasons for allowance were that "none of the cited prior art shows an arrangement of the steps/means for dividing, storing, rendering and displaying in order to form a method and system for enabling a single frame buffer for simultaneous rendering and display in a computer image generator as now claimed." The issue fee was paid on July 22, 2003.

A first Office Action mailed on May 11, 2005 was received from Examiner Vo. He advised that the Notice of Allowance was vacated and that the allowability of claims 1, 3, 4, 15, 17-20 and 22-23 was withdrawn in view of the newly discovered reference to Kajiya et al, U.S. Patent No. 5,864,342. Claims 10-14 were allowed. Claims 2, 5-9, 16 and 21 were objected to as being dependent upon a rejected base claim. Claims 1, 3, 4, 15, 17-20, 22 and 23 were rejected under 35 U.S.C. § 102(e) as being anticipated by Kajiya.

Appellants submitted a response to the Patent Office on August 11, 2005. In this response, Appellants argued that Kajiya teaches a method for using a double frame buffer to reduce the amount of depth memory used (Kajiya, Col. 10, lines 47-60), while the present invention includes the limitation of "**a single pixel frame buffer** for simultaneous rendering and display in a computer image generator."

A second Office Action mailed on February 10, 2006 was received from Examiner Vo. The same claims were allowed, objected to, and rejected as in the previous Office Action. The Office Action was made final. The Examiner asserted that Kajiya teaches a method and system for rendering graphical objects to image chunks utilizing a single pixel frame buffer, citing Kajiya, Col, 6, lines 15-19 for support.

Appellants submitted a response to the Patent Office on April 21, 2006. In this response Appellant amended the claims such that claims 2, 16, and 21 were cancelled and rewritten in independent form including all of the limitations of the base claims 1, 15 and 20 respectively. Claims 10-14 had been allowed. Therefore, Applicant submitted that claims 1, 3-15, 17-20, and 22-23 were placed in a condition for allowance.

An advisory action was mailed on May 25, 2006 from Examiner Roberta Prendergast. Rewriting the dependent claims 2, 16 and 21 in independent form including all of the limitations of the base claims 1, 15 and 20 respectively was deemed to require additional searching. Therefore, the amendment was not entered.

Appellants submitted an after final response to the Patent Office on July 7, 2006. All claims were cancelled without prejudice except for claims 10-14 to put the claims in a condition for allowance.

A fourth Office Action mailed August 28th, 2006 was received from Examiner Prendergast. Claims 10, 11, and 13 were rejected under 35 U.S.C. § 102(e) as being anticipated by Kajiya. The Examiner asserted that Kajiya teaches a method for enabling a single pixel frame buffer for simultaneous rendering and display in a computer image generator, stating that the single rasterization buffer cited in Kajiya at Column 6, lines 15-29 is understood to be a single pixel frame buffer. Claim 12 was

rejected under 35 U.S.C. 103(a) as being unpatentable over Kajiya in view of Grigor et al., U.S. patent No. 6,853,381. The Examiner stated that Kajiya does not specifically teach the step of reducing the transport delay and allowing the display step to overlap a rendering envelope. The Examiner asserted that Grigor teaches this limitation, citing the Abstract, FIGs 1 and 8, column 3, lines 13-33 and column 8, lines 32-67. The Examiner stated that these references disclose that individual display lines comprising a plurality of pixels are rendered and stored in the frame buffer in a memory location only if the location has been previously accessed for display.

Appellants submitted a response to the Patent Office on November 28, 2006. After attempting to put the claims in a condition for allowance four separate times, which eventually resulted in cancelling all claims except 10-14, Appellants reinstated claims 1-9 and 15-23 as claims 24-38. A detailed discussion distinguishing the present invention from Kajiya was included in the response. Regarding the use of a single pixel frame buffer, Appellant argued that Kajiya does not disclose this limitation. The only reference made in Kajiya to a single buffer (Kajiya, Col. 6, lines 15-19), which has been repeatedly relied upon by the Examiners in the continued rejections, is a "single rasterization buffer" cited in the summary of the patent. In the next paragraph of Kajiya, it is explained that the single rasterization buffer is a double buffer. Kajiya does not teach or suggest the idea of using a single pixel frame buffer, but describes on several occasions the use of a double-buffering system, as is standard in the art. Therefore, Appellant argued that the correct interpretation of the cited reference in the patent summary (Col. 6, lines 15-19) is that the system in Kajiya can use a single rasterization buffer that is double buffered.

A fifth Office Action mailed February 27, 2007 was received from Examiner Prendergast. Claims 10-14 and 24-38 were rejected. Claims 25, 30, 31, 33 and 37

were rejected under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which Applicant regards as the invention. Claims 10, 11, and 13 were rejected under 35 U.S.C. 102(e) as being anticipated by Kajiya. The Examiner again argued that Kajiya teaches a method for enabling a single pixel frame buffer, citing column 6, lines 15-29 and asserting that the single rasterization buffer is understood to be a single pixel frame buffer. Claim 12 was again rejected under 35 U.S.C. 103(a) as being unpatentable over Kajiya in view of Grigor. Claims 24-38 were rejected under 35 U.S.C. 103(a) as being unpatentable over Kajiya in view of Taraci et al., U.S. Patent No. 6,316,974.

Appellants submitted a response to the Patent Office on June 26, 2007. Claims 25, 30, 31, 33 and 37, that had been rejected under 35 U.S.C. § 112, were cancelled without prejudice. Claims 10, 32, 34 and 35 were amended to correct for antecedent basis issues. The amendments were made to put the claims in a better form for consideration on appeal under 37 C.F.R. § 1.116. After realizing that little to no progress had been made in the prosecution of the present invention over five separate Office Actions and responses, it was determined that it was in Appellant's best interest to file an appeal. The Notice of Appeal was filed concurrently with the response on June 26, 2007.

An Advisory Action mailed July 5, 2007 was received from Examiner Prendergast. The Advisory Action stated that the 35 U.S.C. § 112 rejection made in the previous Office Action had been withdrawn and the amendments were entered. The rejections 10, 11 and 13 and 12, 14, 24, 26-29, 32, 34 and 35 under 35 U.S.C. 102(e) and 103(a), respectively, remained.

B. Appellants' invention

Reducing Lag Time

Appellants' invention is directed to a method for reducing transport delay in an image generator. The present invention is useful in real-time image generation of computer generated images used in fields such as simulators used to instruct and train operators such as pilots and drivers. Transport delay is the time between when a stimulus occurs, such as a pilot moving a control stick in a simulator, until the last pixel of the generated image is drawn on a display. In other words, it is the lag that occurs between a user instructing a computer generated image to change and when the image has completed updating.

In the real world, high performance vehicles such as cars and jet planes respond rapidly to a user's commands. In the virtual world of simulators, it takes a set amount of time for a computer to regenerate an updated image based on the user's input. For example, when a fighter pilot directs his plane in a new direction the computer must re-generate a new image based on the new course the plane. If the lag time, or transport delay is not sufficiently short, the delay between the user's action and the reaction displayed on the screen can cause motion sickness. This phenomenon is commonly referred to as simulator sickness.

A standard graphics pipeline architecture used in computer graphics generation for real-time simulation is illustrated in FIG. 4 of the application and reproduced below.

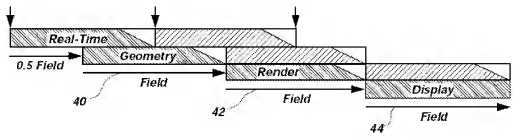


Fig. 4
(PRIOR ART)

The dark shaded boxes represent the flow of one field of data through the graphics pipeline. Each box has a length corresponding to an amount of time it takes to display an image on a screen. At a standard refresh rate of 60 Hertz, a new image can be displayed in approximately $1/60^{th}$ of a second, or 16.67 milliseconds (ms).

The Real-Time box illustrates the time it takes for a real-time controller to receive information from a user, such as a change in direction of a simulated vehicle, and compute the matrices and other information needed to display the scene. A typical time is approximately half a field, or 8.33 ms. The real-time calculations are sent to a geometry processing engine 40 that processes the primitives in each scene. A double buffered memory, referred to as the geometry buffer, is typically used by the prior art between the geometry and rendering 42 processes. This provides one full field time for geometry calculations 40 and the next field time for pixel rendering 42. A second double buffered memory is used by the prior art between the rendering process and the display process 44, typically referred to as a pixel frame buffer. Once the new image has been rendered and written into one side of the double buffered pixel frame buffer, the buffer will be ready to toggle, or swap, at the beginning of the next graphics frame, or field time.

Thus, as illustrated in FIG. 4, a typical lag time between a user's action and the refreshing of the computer generated image on the screen is about 3.5 fields, or about 58.33 ms at a 60 Hertz refresh rate.

The present invention provides methods for enabling a single pixel frame buffer for simultaneous rendering and display in a computer image generator. As illustrated in FIG. 12 of the Application, which has been reproduced below, a single buffered pixel frame buffer can enable a portion of the rendering process 144 and the display process 142 to occur concurrently.

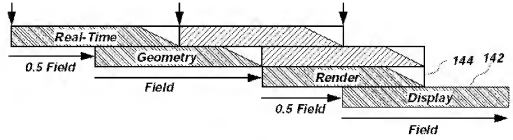


Fig. 12

For example, a portion, such as a single image line, or half of the lines in a computer generated image, can be rendered and stored into the single pixel frame buffer. The display process can then begin retrieving the stored information from the single pixel frame buffer and displaying rendered pixels on the simulator screen. The portion of the image that is rendered and stored in the single buffered pixel frame buffer depends on a difference in time it takes for the rendering process and the display process to occur. If the two processes occur in a relatively similar amount of time, a single line of an image may be stored in the single buffered pixel frame buffer before the display process retrieves the information. If there is more disparity

between the speeds off the two processes, more delay, such as rendering of half the screen, may be needed.

The amount of time that can be saved by using a single pixel frame buffer is significant. At least half of a field time can be shaved off of the total lag time by concurrently running the rendering process 144 and the display process 142 together for at least half a field. Using the previous example, at a display rate of 60 Hertz, the lag time can be reduced from 58.3 ms to 50 ms, a 14% reduction in lag time. If the rendering and displaying processes are closer in time, nearly an entire field of time can be saved, allowing the lag time to be reduced to just over 2.5 fields or 41.67 ms for a 28% reduction in lag time. Thus, the present invention provides a significant improvement in the real-time generation of images by reducing lag time to increase simulator performance and reduce cases of simulator sickness.

Chunking Architecture

In order to render high-quality scenes, many systems store not only the color of each pixel, but compute the color for multiple sub-pixels within each pixel for anti-aliasing. Along with the color values, depth values are computed and stored in order to determine which primitives are in front and which are behind. This sub-pixel frame buffer requires a lot of memory, and more importantly, a lot of memory bandwidth. Memory bandwidth is the speed at which memory can be accessed. Real-time systems typically require rapid access. Thus, high-quality and high-performance systems tend to be very expensive due to the need for large amounts of the fastest types of memory available.

"Chunking" architectures have been used for many years as a way to reduce the cost of memory associated with real-time graphics generation. The Assignee of the present invention has been using chunking architectures since the late 1980's. As

evidence, a portion of a user manual of a system sold by the Assignee entitled "ESIG-3000/2000 System Overview" and published on June 3, 1991, is included in the Section X Evidence Appendix. The user manual discloses a system that includes a Clipper Section (see page 6) in which the display screen is divided up into a uniform grid of 32 x 32 pixel areas called macro spans. These macro spans are equivalent to the 32 x 32 pixel chunks that will be described below in the Kajiya prior art reference. Chunking architectures are used to reduce the amount of memory needed in a computer graphics device since memory is typically one of the more expensive components in the computer graphics device.

In a typical chunking architecture, the amount of memory is reduced by dividing a screen into smaller areas. Rather than store the sub-pixels for the entire screen, the sub-pixels are only stored for one "chunk" of the screen. A chunk is a rectangular block of pixels often about 32x32 pixels in size. The "chunk buffer" is a mini-frame buffer that stores the sub-pixel depth and color values for the pixels within the chunk. After performing the anti-aliasing operations, a single resolved pixel color is then stored in the full screen frame buffer. Thus the chunking architecture removes the need for very large, high bandwidth memory frame buffers and replaces it with a smaller high bandwidth chunk buffer which is much less costly to build.

One disadvantage of this architecture is that the primitives must be sorted into bins associated with each chunk. This bin memory is also double buffered, and therefore adds one more frame of latency to the rendering process. A flow chart illustrating a typical chunking architecture is illustrated in FIG. 5 of the present application and has been reproduced below.

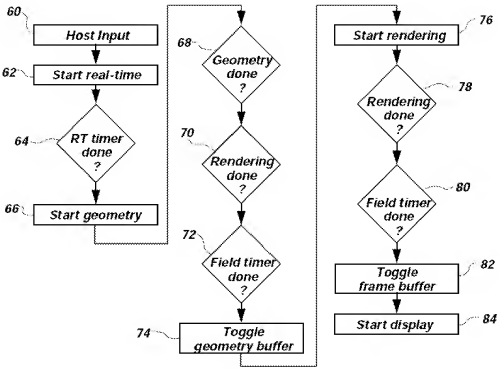


Fig. 5
(PRIOR ART)
"Chunking Architecture"

At the beginning of each new frame, both the frame buffer and the bin memory are toggled. There are now three distinct operating domains; the first is the binning process, second is the rendering operation, and finally the display process. This illustration correlates with FIG. 4 displayed above. As in FIG. 4, the real time process 64 illustrated in FIG. 5 would take approximately half of a frame time, resulting in an overall lag time of approximately 3.5 frames.

Each of the primitives within the scene must be tested against the bins associated with each "chunk" of the screen. The primitives are thereby sorted in to screen related bins. The present invention provides that each primitive that is on screen will be stored in one or more bins. The basic rendering operations are the same as in the

traditional architecture discussed above, but rather than render each primitive to completion, only the portions of the primitives overlapping the current bin are rendered. One bin at a time is rendered, but the bins can be rendered in any order.

Once all of the primitives touching one bin are rendered, the "chunk" buffer can be toggled, the results anti-aliased, and the resolved pixel color can be stored in the full screen frame buffer.

Reduced Lag Time with Chunking Architecture

As previously discussed, the extra frame of delay introduced by the binning process can be problematic in real-time image generation. By selecting a top-to-bottom order of rendering the bins, and by using a single buffer frame buffer, it is possible to remove the extra frame of delay. A flow chart depicting one embodiment of the present invention is illustrated in FIG. 13 and has been reproduced below.

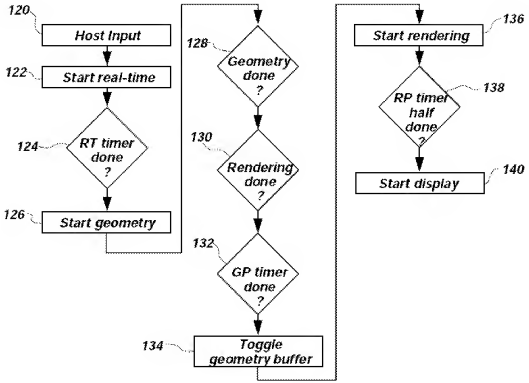


Fig. 13

Reduced Latency Chunking Architecture

As in the previous example, the bin memory is toggled at the beginning of each field. However, in the reduced latency chunking architecture illustrated in FIG. 13 there is not a double buffered frame buffer to toggle. With a standard single buffered frame buffer, primitives are typically updated as they are drawn. With the method disclosed in the present application that distracting artifact is avoided by rendering at least one full row and then displaying that row. The bins are rendered in a top-to-bottom order. The order of bins within a given row is irrelevant, but the top row is completely rendered before the next row begins. Once a complete row has been rendered, that row can then be displayed without waiting for the entire scene to be rendered.

For most display devices, the flow of pixels must be a steady stream without interruption. Thus, care is typically taken to insure that each new row of chunks can be rendered before the previous row has been displayed. Since the rendering complexity is often not uniform on screen, it may be necessary to buffer up multiple chunk rows before starting the display process. FIG. 13 illustrates an embodiment wherein half of the rows are rendered prior to the display process. Rendering the bins for the first half of the rows in a computer generated graphical image provides a larger head start that allows for greater differences in the time it takes to render the bins and place them into the single pixel frame buffer, and the time it takes to display the rendered bins.

A single buffering process requires additional complexity that is not required when a double buffer is used. A double buffer allows an entire display frame to be rendered and stored on one side of the double buffer, then toggled to the other side of the double buffer for displaying. Thus, the double buffer eliminates potential interference between the rendering process and the display process if the two processes occur at different speeds. With a single buffer, if the rendering process and display process occur at different speeds, extra care must be taken to ensure that one process does not encroach on the other.

Since the frame buffer is single buffered in the present invention, if the rendering operations fall too far behind the display process, portions of an old frame will be redisplayed and produce an erroneous picture. Alternatively, if the rendering operations get too far ahead of the display process, the newly rendered data stored in the single buffered frame may overwrite sections that have not been displayed yet. Thus, appropriate measures can be taken to keep the two independent processes from bumping in to each other.

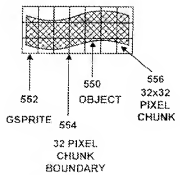
C. The Asserted References

1. The Kajiya Reference

Kajiya discloses a method for reducing the amount of processing required to update a computer generated graphical image by only updating altered portions of an image when a change in view point has occurred by transforming the object geometry of the altered portions using an affine transform based on the change in view point. (See Col. 14, line 43 to Col. 15, line 37).

Kajiya uses a chunking method that involves dividing a scene into non-interpenetrating objects 550 referred to as gsprites 552, as illustrated in FIG. 15B of Kajiya which has been reproduced below. The gsprites are then divided into rectangular chunks 556 such as 32 x 32 pixel regions. (See Col. 10, lines 35-42; Col. 32, lines 30-35).

FIG. 15B



Kajiya discloses a process used to reduce transport delay between a user instructing a computer generated image to change, and when the image has completed updating. The process disclosed in Kajiya takes advantage of the affine transform in combination with the chunking architecture disclosed in Kajiya to warp gsprites that have changed due to a change in viewpoint. (See generally, Col. 56, line 18 to Col.

57, line 34). The process disclosed in Kajiya is significantly different from the claimed invention in the present application.

The process to reduce transport delay disclosed in Kajiya involves using viewpoint data from a subsequent image to affine transform, or warp affected objects or gsprites. (Col. 56, lines 41-51). An affine transformation matrix is computed based on the modeling transform for the current image and the subsequent image. This enables a quick adjustment when rapid changes in viewpoint perspective of a user occur. (Col. 56, lines 52-60).

Kajiya discloses in more detail the actual implementation of the affine transform display method to reduce transport delay. (See generally, Col. 58, line 24 to Col. 62, line 16). Gsprite chunk image data is rendered and affine transforms are calculated for a frame. (Col. 59, lines 62-67). A gsprite engine maps the gsprite image to output device coordinates and transforms the gsprite data. Pixel data is then sent to a double buffered compositing buffer that allows pixel data to be transferred from one buffer while pixel data is being composited in the other buffer. (Col. 60, lines 1-7).

Kajiya discloses another embodiment in which a display device is divided into bands. The term "band" refers to the amount of time (band period) allotted to process a band of pixel data. The time can be derived, in part, from the frame rate and the number of bands in the display device. (Col. 60, lines 54-59). While the gsprite engine fills the compositing buffer for one band, the compositing buffer transfers composited image data for another band to a digital to analog converter. In the next band period, the band that was composited is then displayed. (Col. 60, lines 60-67). The two compositing buffers ping-pong back and forth so that as one scanline region is being displayed, the next is being composited. (Col. 61, lines 5-13).

2. The Grigor Reference

Grigor discloses a write behind controller configured to receive information from a display device controller in order to determine a current location available in a frame buffer for receiving information. Write access to the frame buffer by a rendering engine is prohibited if the access is to an area below a currently available location of the frame buffer. The rendering engine is stalled when the requested address location has not yet displayed its data. Subsequently, the write access to the frame buffer is allowed when the address has been rastered. (See Col. 8, lines 22-60; Abstract).

3. The Taraci Reference

Taraci teaches a method and apparatus for vertically locking input and output video frame rates from digital video sources to eliminate mixing of pixels from different input frames having different resolutions and scan rates into one output frame. This is accomplished using two phase locked loops (PLLs) that are connected in series. (See Col. 13, lines 16-47). The use of two PLLs provides the ability to lock output and input vertical sync pulses by adjusting a reference frequency to the pixel clock generating PLL. Applications include graphics switching, scan conversion, video scaling, line doubling, and line quadrupling. (Col. 18, lines 12-31).

C. Rejections Under 35 U.S.C. § 102(e)

1. Requirements for anticipation

The Examiner has rejected pending claims 10, 11 and 13 under § 102(e) as being anticipated by Kajiya. It should be noted that Kajiya issued as a patent on January 26, 1999, some 21 months prior to the filing of the present application on October 23,

2000. Thus, it appears a rejection under 35 U.S.C. 102(b) would be accurate. Regardless, the analysis of the §102 rejection is the same whether the rejection is based on §102(e) or §102(b). The issue at hand is whether the invention was fully described in the Kajiya patent.

2. The rejection of Claims 10, 11 and 13 as being anticipated by Kajiya

According to MPEP §706.02 (IV), the reference must teach every aspect of the claimed invention either explicitly or impliedly. Any feature not directly taught must be inherently present. Appellants submit that the Examiner has failed to show that Kajiya teaches every aspect of the claimed invention. The Examiner has repeatedly relied on a reference to the summary in Kajiya, Col. 6, lines 15-29 as evidence that Kajiya teaches the use of a single pixel frame buffer. As Appellants have argued continually, the elements disclosed in Kajiya do not correspond to those of Appellants' invention.

Independent claim 10 of the present application reads as follows:

10. A method for enabling a single pixel frame buffer for simultaneous rendering and display in a computer image generator, comprising the steps of:

(a) dividing a geometry buffer into a plurality of screen bins;

(b) storing primitives in each screen bin the primitives touch;

(c) rendering the screen bins by row from top to bottom, into the single pixel frame buffer;

(d) displaying at least one row of screen bins rendered before the rendering of all the screen bins has completed, wherein the displaying of the screen bins takes place after a selected portion.

As previously discussed, Appellants invention uses a single pixel frame buffer configured for simultaneous render and display of computer graphics image

information in a computer image generator. The single pixel frame buffer enables a reduced transport delay by reducing the number of frames of delay in the computer image generator between a user instructing a computer generated image to change, and when the image has completed updating.

Buffers are typically used to accommodate delay between different processes. Double frame buffers allow a first process to write to a first frame of the buffer. The contents of that frame can then be toggled to the second frame of the double frame buffer, which can be read by a second process. In contrast, if a single frame buffer is used, the same buffer is used for writing by the first process and reading by the second process. Thus, the use of a single frame buffer causes added complexity since it must be ensured that the first process does not overwrite data that has not yet been read by the second process, or that the second process does not read the same data twice that has not been updated by the first process.

In the present invention, a single pixel frame buffer is used between the rendering and displaying processes to reduce the added delay associated with the use of a double frame buffer. The present invention discloses the use of a hardware interlock which can be used to ensure that the rendering process and displaying process do not interfere with each other when using a single buffer between the processes. (See Application, page 14, lines 3-8).

The Examiner has repeatedly cited the summary of Kajiya, Col. 6, lines 15-29 as evidence that Kajiya teaches the use of a single pixel frame buffer. In the Final Office Action mailed February 27, 2007, the Examiner states that the single rasterization buffer recited in this citation is understood to be a single pixel frame buffer." (See Page 4). The entire citation in Kajiya reads:

> The geometry for a chunk can be rendered to a common depth buffer or rasterization buffer. For example, all of the geometry for a first chunk can be rasterized into a single rasterization buffer before geometry for the next chunk is rasterized to the same rasterization buffer. Similarly, pixel data generated while rasterizing primitives for a chunk, including pixel fragments to support high-quality anti-aliasing and translucency computations, can be resolved for a first chunk before the next chunk is resolved. Using a common depth or rasterization buffer saves memory. Moreover, the process of resolving each chunk in this manner provides an effective form of compression because pixel fragments need not be stored before an entire frame of image data is generated.
>
> **The rasterization buffer can be double-buffered** such that the geometry for a first chunk is rasterized to generate pixel data, while the pixel data for another chunk is being resolved. This capability enables sophisticated anti-aliasing and translucency computations to be performed without adding delay in the rendering pipeline. (Col. 6, lines 15-34; emphasis added)

A brief overview of the process disclosed in Kajiya will be helpful in understanding what the rasterization buffer is and how it is used. The graphical image display process disclosed in Kajiya is substantially more complex than the process disclosed in the present application. Kajiya not only renders graphical images, but also performs affine transformations on the rendered images to reduce the amount of computing needed to display the images as the viewpoint changes. The rasterization buffer is part of an image processor 462 for producing rendered image data from geometric primitives, as illustrated in FIG. 4b of Kajiya. Kajiya explains that the rasterization buffer is not a single buffer, but a group of buffers that includes pixel buffers 472 and a fragment buffer 470. The rasterization buffer supports double

buffering of pixel data and single buffering of fragment data. The fragment data occurs due to the gsprite chunking process used in Kajiya. (Col. 61, lines 37-58).

FIGs. 9a-9c of Kajiya illustrate that the rasterization buffer (comprising the fragment buffer(s) 410 and the pixel buffers 408) are part of the Tiler 200. The Tiler 200 is shown in FIG. 4a, located at the beginning of the rendering process. Graphical information is sent to the Tiler 200 to render the images. The rendered data is then sent to the gsprite engine 204 where warping using affine transformations occurs. Information from the gsprite engine 204 is then sent on to post-processing 210 where alpha (opacity) and color buffers are implemented, and finally the digital data is sent to a digital to analog converter (DAC) 212 before being sent to a display device.

In contrast, the single pixel buffer in the present invention is placed between the rendering process 144 and the display process 142, as illustrated in FIG. 12. No warping or affine transformation occurs in the present invention. The single pixel buffer enables a single row of a graphical display to be rendered into the buffer before the display process scans the rendered row. (See specification, page 11, lines 3-14). The transport delay can be reduced using the single pixel frame buffer, which can be accessed by both the rendering and display processes. (Page, lines 10-19).

The single pixel buffer, as used in the present application, is more akin to the compositing buffer disclosed in Kajiya. The compositing buffer is used to collect and hold information ready for display until it is displayed. However, the compositing buffer in Kajiya is a traditional double buffer where transformed chunks are composited together to form an image. (See Kajiya, Col. 60, line 59 to Col. 61, line 13). Kajiya does not take advantage of a single buffer compositing buffer to reduce transport delay. There is no mention in Kajiya of a single buffer that is used to reduce transport delay.

Additionally, the compositing buffer is not used in Kajiya for simultaneous rendering and display of a computer image, as the single pixel buffer is in the claimed invention. Rather, rendering of the image occurs in the Tiler 200 (FIG. 4a), as previously discussed. The image is then sent into the gsprite engine 204 where the chunks are warped using the affine transform. The warped chunks are then sent through post processing and then composited and stored in one frame of the double buffered compositing buffer, then toggled to the other buffer in the compositing buffer that can be accessed for displaying the graphical information.

The single reference to a "single rasterization buffer" in the summary of the Kajiya patent is not synonymous with the single pixel frame buffer disclosed in the present application and recited in independent claim 10. The rasterization buffer disclosed in Kajiya is a group of buffers, including the double buffered pixel buffers. Kajiya does not disclose the use of a single buffer compositing buffer, a single buffer rasterization buffer, or any type of single buffer used to reduce transport delay. Nor does Kajiya describe the added complexity that would be needed to enable two separate processes to read and write to the same memory to allow simultaneous rendering and display, as previously described.

Therefore, Appellants respectfully submit that independent claim 10 presents patentable subject matter, and that the rejection of this claim should be overturned.

Rejection of the dependent claims 11 and 13 should also be overturned for at least the reasons given above with respect to the independent claim. The dependent claims, being narrower in scope, are allowable for at least the reasons for which the independent claim is allowable.

D. Rejections Under 35 U.S.C. § 103(a)

1. Requirements for Prima Facie obviousness

The Examiner has rejected all of the pending claims under § 103(a) as being *prima facie* obvious over a number of references. The Patent and Trademark Office (PTO), through the Examiner, has the burden of establishing a *prima facie* case of obviousness. *In re Fine*, 837 F.2d 1071, 5 U.S.P.Q.2d 1596, 1598 (Fed. Cir. 1998). To satisfy this burden, the PTO must meet the criteria set out in M.P.E.P. § 706.02(j):

> [T]hree basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art and not based on applicant's disclosure. *In re Vaeck,* 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991).

Moreover, the obviousness analysis must comply with the statutory scheme as explained by the Supreme Court in *Graham v. John Deere Co.*, 383 U.S. 1, 17 (1966), namely, consideration must be given to: (1) the scope and content of the prior art, (2) the differences between the prior art and the claimed invention, (3) the level of ordinary skill in the pertinent art, and (4) additional evidence, which may serve as indicia of non-obviousness.

In order to combine references, the prior art must provide some reason or motivation to make the claimed compositions, *In re Dillon*, 16 U.S.P.Q.2d 1897, 1901 (Fed. Cir. 1990). As aptly stated in *In re Jones*, 21 U.S.P.Q.2d 1941, 1943-44 (Fed. Cir. 1992):

> "Before the PTO may combine the disclosure of two or more prior art references in order to establish *prima facie* obviousness, there must be some suggestion for doing so, found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art... Conspicuously missing from this record is any

> *evidence*, other than the PTO's speculation (if it be called evidence) that one of ordinary skill in the…art would have been motivated to make the modifications of the prior art necessary to arrive at the claimed (invention)."

An excellent summary of how the prior art must be considered to make a case of *prima facie* obviousness is contained in *In re Ehrreich et al.*, 220 U.S.P.Q. 504, 509-511 (CCPA 1979). There the court states that a reference must not be considered in a vacuum, but against the background of the other references of record. It is stated that the question of a § 103 case is what the reference(s) would "collectively suggest" to one of ordinary skill in the art. However, the court specifically cautioned that the Examiner must consider the entirety of the disclosure made by the reference and avoid combining them indiscriminately.

In finding that the "subject matter as a whole" would not have been obvious in *Ehrreich* the court concluded:

> "Thus, we are directed to no combination of prior art references which would have rendered the claimed subject matter as a whole obvious to one of ordinary skill in the art at the time the invention was made. The PTO has not shown the existence of all the claimed limitations in the prior art or any suggestion leading to their combination in the manner claimed by applicants." (underlining added)

It is true that an obviousness determination is not the result of a rigid formula disassociated from the consideration of the facts of a case. Indeed, the common sense of those skilled in the art demonstrates why some combinations would have been obvious where others would not. See KSR Int'l Co. v. Teleflex Inc., 550 U.S. __, 2007 WL 1237837, at *12 (2007). However, it has been widely recognized that virtually every invention is a combination of elements and that most, if not all, of these will be found somewhere in an examination of the prior art. This reasoning lead the court, in *Connell v. Sears, Roebuck & Co.*, 220 U.S.P.Q. 193, 199 (Fed. Cir. 1983) to state:

> "...it is common to find elements or features somewhere in the prior art. Moreover, most if not all elements perform their ordained and expected function. The test is whether the claimed invention as a whole, in light of all the teachings of the references in their entireties, would have been obvious to one of ordinary skill in the art at the time the invention was made." (underlining added)

With the above background in mind, Appellants contend that the Examiner has not met this burden with respect to the claims rejected under § 103. Particularly, Appellants submit that the PTO has failed to show that each and every element of the claimed invention is contained in the combined references, and that there was sufficient reason to modify the asserted prior art references to achieve the present invention. Appellants now turn to a discussion of the individual rejections at issue, and the references on which they are based.

2. The rejection of Claims 12 and 14 over Kajiya in view of Grigor

Rejection of the dependent claims 12 and 14 should also be overturned for at least the reasons given above with respect to independent claim 10. The dependent claims, being narrower in scope, are allowable for at least the reasons for which independent claim 10 is allowable.

3. The rejection of Claims 24, 26-29, 32 and 34-35 over Kajiya in view of Taraci

According to M.P.E.P. § 706.02(j), to render a claim *prima facie* obvious, the asserted prior art reference (or references when combined) must teach or suggest all of the claim limitations. Appellants submit that the combinations asserted by the Examiner does not teach or suggest each and every element of the rejected claims.

As Appellants have argued repeatedly, the elements of the method taught in Kajiya do not correspond to those of Appellants' invention.

As previously discussed, the Kajiya reference fails to teach or suggest the use of a single buffer for simultaneous rendering and display in a computer image generator. Independent claims 24 and 32 both include the limitation of rendering screen bins into a single pixel frame buffer to enable simultaneous rendering and display.

Additionally, both independent claims 24 and 32 include the limitation of displaying at least one rendered screen bin when the rendering of the screen bins for the single pixel frame buffer is at least ½ completed. The Final Office Action mailed February 27, 2007 states that Kajiya does not specifically teach this limitation. (See page 8). However, the Examiner asserts that Taraci teaches the limitation that an output/display vertical frame read pointer is placed at a point in reference to an input/render vertical write pointer to create a frame rate delay of ½ a frame indicating that rendering is at least ½ completed before displaying begins. The Examiner cites Taraci, Col. 8, lines 30-45 to support her assertion.

Appellants respectfully disagree with the Examiner's characterization of Taraci. The citation to Taraci is in the background section of the patent. The background section discusses problems that occur with state of the art seamless graphics switchers (SGS) that are capable of switching between multiple analog and digital input formats and resolutions while keeping the output rate and resolution stable. (See Col. 4, line 65 to Col. 5, line 6).

Taraci discusses problems that occur in SGS caused by frame delays at various points in a video system. (Col. 8, lines 15-25). Taraci then discusses attributes of an SGS that would be desirable. The Examiner's citation states that:

> For example, a desirable SGS is one where the read and the write pointers in memory do not cross, and it does not produce an output frame made up of two different input frames. It is also desirable to provide a system wherein the output and input vertical sync pulses are locked, and the position of the output vertical frame read pointer in memory can be adjusted as compared to the position of the input vertical frame write pointer. For instance, a desirable SGS would allow the output read pointer to be placed at any point in reference to the input vertical pointer, so that for instance, frame rate delay could be adjusted (e.g. to say, half a frame). Thus, frame rate delay for an SGS could be set at, adjusted to, or programmed to change to one or more constant, predictable values as desired. (Col. 8, lines 30-45).

While it may be desirable to allow a frame rate delay for a seamless graphics switcher to be adjusted, Appellants do not believe that this citation, or anywhere in Taraci, teach or suggest the limitation of displaying at least one rendered screen bin when the rendering of the screen bins for the single pixel frame buffer is at least ½ completed. Therefore, the combination of Kajiya and Taraci would not result in all of the limitations of independent claims 24 and 32. Rather, the combination of Kajiya and Taraci would result in a graphics generator configured to affine transform chunks, composite the chunks in a double buffered compositor buffer, and double the number of lines of resolution at the output of the buffer. Of course, the additional signal processing and buffering disclosed in Taraci would result in substantial additional delays in the transport delay. Thus, the combination of Kajiya and Taraci teaches away from the present invention, which reduces transport delay by enabling the simultaneous rendering and display in a computer image generator using a single pixel frame buffer.

For the reasons stated above, Appellants respectfully submit that independent claims 24 and 32 present patentable subject matter, and that the rejection of these claims should be overturned.

Rejection of the dependent claims 26-29 and 34-35 should also be overturned for at least the reasons given above with respect to the independent claims. The dependent claims, being narrower in scope, are allowable for at least the reasons for which the independent claims are allowable.

CONCLUSION

Appellants respectfully submit that the claims on appeal set forth in the Appendix are patentably distinct from the asserted prior art references. Particularly, none of the asserted references or combinations of references motivates, teaches, or suggests one of ordinary skill in the art within the meaning of 35 U.S.C. § 102(b) or 35 U.S.C. § 103(a) to arrive at the presently claimed invention. Appellants contend that neither Kajiya alone nor Kajiya in combination with Taraci and/or Grigor teach each and every element of the claimed invention, and furthermore that they provide no reason to combine them.

For these reasons, Appellants respectfully request that the Board of Appeals reverse the rejection and remand the case to the Examiner for allowance.

Dated this 27th day of August, 2007:

/Steve M. Perry/

Steve M. Perry
Attorney for Applicant
Registration No. 45,357

THORPE NORTH & WESTERN, LLP
8180 South 700 East, Suite 200
Sandy, Utah 84070

(801) 566-6633

On Behalf Of:
Rockwell Collins, Inc.
400 Collins Rd. N.E.
Cedar Raids, IA 52498

VIII. CLAIMS APPENDIX

1-9 (canceled)

10. (previously presented) A method for enabling a single pixel frame buffer for simultaneous rendering and display in a computer image generator, comprising the steps of:

(a) dividing a geometry buffer into a plurality of screen bins;

(b) storing primitives in each screen bin the primitives touch;

(c) rendering the screen bins by row from top to bottom, into the single pixel frame buffer;

(d) displaying at least one row of screen bins rendered before the rendering of all the screen bins has completed, wherein the displaying of the screen bins takes place after a selected portion of the screen bins for a current field have been rendered.

11. (original) A method as in claim 10 further comprising the step of reducing the transport delay without allowing the display step to overlap a rendering envelope.

12. (original) A method as in claim 10 further comprising the step of reducing the transport delay and allowing the display step to overlap a rendering envelope.

13. (original) A method as in claim 10 further comprising the step of rendering at least one row of screen bins before the display step begins.

14. (original) A method as in claim 10 further comprising the step of reducing the transport delay by allowing the display step to overlap a rendering envelope without allowing pixels from a previous field to be displayed.

15-23. (canceled)

24. (previously presented) A method for enabling a single pixel frame buffer for simultaneous rendering and display in a computer image generator, comprising the steps of:

(a) dividing a geometry buffer into a plurality of screen bins;

(b) storing primitives in each screen bin containing a portion of the primitive;

(c) rendering the screen bins, by row from top to bottom, into the single pixel frame buffer;

(d) displaying at least one rendered screen bin when the rendering of the screen bins for the single pixel frame buffer is at least ½ completed.

25. (canceled)

26. (previously presented) A method as in claim 24, further comprising the step of using a hardware interlock to ensure that the rendering step does not advance ahead of the display step.

27. (previously presented) A method as in claim 26, further comprising the step of using a row based hardware interlock to ensure that the rendering step does not advance ahead of the display step.

28. (previously presented) A method as in claim 24, further comprising the step of executing the rendering and displaying steps concurrently within the same frame buffer.

29. (previously presented) A method as in claim 24, wherein step (d) further comprises using an independent timer to control toggling of the geometry buffer.

30. (canceled)

31. (canceled)

32. (previously presented) An image generator with a single pixel frame buffer enabled for simultaneous rendering and display, comprising:

(a) a geometry buffer divided into a plurality of screen bins;

(b) a plurality of primitives, stored in all of the screen bins that touch a screen region defined by the screen bin;

(c) a rendering engine, configured to render the primitives in the screen bins into the single pixel frame buffer by row and from top to bottom;

(d) a display processor, configured to display at least one rendered screen bin on a display screen before the rendering engine has completed rendering all the screen bins; and

(e) wherein the display processor begins to display the screen bins rendered when the rendering of the screen bins is at least ½ complete.

33. (canceled)

34. (previously presented) An image generator as in claim 32, further comprising a geometry engine configured to transform a database and the plurality of primitives used by the image generator.

35. (previously presented) An image generator as in claim 32, further comprising a real-time controller configured to receive real-time control information and compute transformation matrices.

36-38. (canceled)

IX. EVIDENCE APPENDIX

I. Moon, "ESIG-3000/2000 System Overview," June 3, 1991

ESIG-3000/2000 System Overview
3-Jun-1991
by
Richard N. Moon

ENVIRONMENT PROCESSOR

The Environment Processor is comprised of a Clock and Environment Memory (CEM) card (230110) and one or more Triple Arithmetic Processor (TAP) cards (230112).

Clock and Environment Memory (CEM) card

The CEM card provides five major functions: first, the interface with the VME unit via the VES bus; second, diagnostics capabilities via serial shift loops and other control registers; third, on-line database storage in the environment memories; fourth, RTS control of the various sections of the Channel Processor via the GO and DONE signals; and fifth, video control for the multiple Video Buffer cards.

All communications between the Image Generator Controller (the MVME147 card) and the Channel Processor must pass through the CEM card. There are two major methods employed on the CEM for communications between the IGC and the Channel Processor. The first is by means of dedicated control registers which are memory mapped into VME address space and can be written and/or read from the IGC. A subset of these registers provide serial shift loop access to hardware elements throughout the channel processor. The primary uses of the serial shift loops are for off-line diagnostics procedures, and for RTS initialization of control rams. However, during realtime operation, the serial shift loops may be used for limited feedback of statistical data from the Macro-poly Generator card and laser range data from the Span Assembler card. The second major method of communication with the Object Manager and Geometric Processor in the channel processor is through the environment memories.

Environment memory (EVM)

Each CEM card contains environment memories which are used for on-line storage of the active data base. The environment memory consists of two banks of memory, each of which is memory mapped into VME address space and is accessible directly by the IGC across the VES bus. The environment memory is arranged in 32-bit word format, and can be accessed by the IGC and the arithmetic processor cards only on full word boundaries. The standard CEM card (230110-100) contains 4 Megabytes of memory in each of the two banks for a total of 8 Megabytes. This standard memory is implemented using 1 Meg-by-one-bit DRAMS. The expanded CEM card (230110-101) contains 4 Meg DRAM parts, providing 16 Megabytes of memory in each of the two banks. The printed circuit card is designed to handle either of the two memory types. However, the only practical upgrade path from the standard CEM card to the expanded CEM card is to exchange cards, since replacement of the memory chips themselves is impractical.

Although both banks of memory are accessed from a single bus (VES bus) on the VME side, each bank has its own independent bus to all arithmetic processors in the environment processor. These output busses are 32-bits wide and are designed to provide maximum output bandwidth. The memory chips are accessed using static column addressing, and a full word can be transferred from EVM to the requesting arithmetic processor card each clock period of 50 nanoseconds. This high speed data transfer can be sustained continuously except for page boundary crossings, VES accesses or for memory refresh cycles.

One bank of memory is referred to as the "mesh memory" and the other is called the "object memory." Mesh memory is used to store the hierarchical structure of the database or the mesh and cell data. The arithmetic processors assigned to provide Object Manager functions deal almost exclusively with data in the mesh memory. Object memory is used to store the on line data base objects. Objects are composed of polygons and lights and their associated parameters . The arithmetic processors assigned to GP functions deal primarily with data in the object memory. This separation of functions and their associated memory provides for efficient utilizations of the data buses and minimizes interference between the two functions.

EVM Capacities

Assuming the average polygon (a smooth shaded polygons of 84 bytes with two vertices of 12 bytes each and object overhead of 4 bytes per polygon) requires 112 bytes, the standard size object memory can store a maximum of 37,000 polygons. However, not all of object memory may be available for polygon storage. For example, 8K bytes are reserved for TBUG and OM and GP microcode load buffers. Other functions, such as texture paging, may require dedicated allocations of object memory.

Assuming the average cell size (one cell at 64 bytes plus two plane equations at 12 bytes each) is 88 bytes, the standard-sized mesh memory would be capable of storing 47,600 cells. However, the practical number of cells which can be accommodated must be less than this, since other data structures reside in mesh memory. Currently, the OM and GP command lists, coordinate system tables, and paging and memory allocation tables (for 8 channel processors and 8 viewports) occupy up to 650K bytes. This overhead leaves room for approximately 40,300 cells.

The expanded CEM card, with its four times larger memories, can store a maximum of 149,000 polygons and 190,000 cells.

Assuming full utilization of the bus from the object memory to the GP cards, approximately 7700 polygons could be transferred for processing in a 60 Hz frame. If 40% of these polygons were potentially viewable, then the bus capacity limits system polygon capacity to a maximum of approximately 3000 polygons.

Triple Arithmetic Processor (TAP) cards

Object management (OM) and geometric processing (GP) functions are performed by a set of arithmetic processor units operating in parallel. Three arithmetic processing units are housed on each TAP card. The standard backpanel for both the ESIG-3000

and the ESIG-2000 has five slots wired to accept TAP cards, providing for a maximum of 15 arithmetic processing units. Normal configurations will contain no more than 3 TAP cards, but expansion is provided for future computationally intensive functions. From the number of arithmetic processing units available in a specific configuration, the user can specify which units are to be dedicated to OM functions and which units are to be dedicated to GP functions. This allocation of processors to OM and GP functions must be made at system initialization time and cannot be dynamically changed during run time operation.

For realtime operation, a TAP card must be present in TAP slot 0, and processing unit zero (in that card) must be assigned to perform GP functions. This assignment is necessary since GP-0 processor serves as the master during lock-step operation.

A maximum of eight processors can be assigned to OM processing. OM processors must reside in TAP cards in the first three TAP slots in the backpanel (i.e. they can be any of processors 1 thru 8). Other than these two restriction just mentioned, assignment of processing units to OM and GP functions can be made in any order and in any combination.

Each processing unit utilizes a Texas Instrument TMS320C30 digital signal processing chip operating with a 50 nanosecond clock. The TMS DSP chip was chosen because it has a fairly large internal data memory, 2K words. Since operation is extremely efficient while internal memory is utilized, both cell processing in the OM and polygon processing in the GP are performed almost exclusively using internal memory. Each TMS chip has 8K words of static memory dedicated to microcode instructions. Each processor also has 64K words of static data memory. In the event that the code memory were to become full, microcode can be stored and executed from the 64K scratch memory with no loss of efficiency, as long as EVM I/O is not involved. The data memory, commonly referred to as scratch memory, is used primarily for data tables such as SINE, COSINE and color values. Each processor also has an output FIFO memory, used to store the processed polygon output data for transfer to the Clipper section of the pipeline.

Microcode is loaded into the program memories for each TMS processor from EVM. A small bootstrap loader is first loaded into the scratch memories of each processor via the serial shift loops. This loading of bootstrap code using the shift loops takes place in parallel for all processors. At the time of bootstrap code loading, each arithmetic processor is designated as either an OM or GP processor. The microcode loader then reads a 1K-word block of GP microcode from disk to EVM and issues a GPGO which causes the bootstrap loaders in those arithmetic processors previously designated as GP processors to download the code from EVM into internal program memory. The GP microcode load process is continued, a block at a time, until complete. The OM microcode is then loaded by a similar process with OMGOs being issued to initiate the download process in OM-designated processors.

A special software microcode debug tool (TBUG) has been written to provide the developer access to his code as it is operating within the TMS chip. This debugger microcode is loaded into each processor in addition to the GP and OM microcode and is retained in code memory for activation at any time. The user interface to this special

debugger has been designed to closely mimic the interfaces used in the TMS320 emulator and VAX software debuggers provided by Texas Instruments. The driver and interface to the microcode debugger is written in "C", executes in the 68030 card and communicates with the debugger microcode through EVM.

Object Manager (OM)

The term Object Manager refers to the functions performed at the mesh and cell level of the hierarchical data base structure. The term is also used to refer to the microcode performing these functions and to the arithmetic processor units assigned to execute OM microcode.

OM functions include the following:

Extrapolation of moving model parameters (position & attitude)
Creation of 3-by-3 rotational matrices from Euler angles or Quaternions
Moving model, static model and viewport coordinate system creation
Database paging functions, including static and dynamic models, terrain and feature database, basis sets and texture
Database tree traversal, including FOV object culling, cellular priority resolution, level of detail determination, and control box determination.
Animation and select switching
Smooth shading and texture computations for basis sets
Multiple light source computations
Height above terrain
Collision detection

A single OM processor has the capacity to compute approximately 1000 cell touches in 16 milliseconds. This throughput, of course, varies greatly with database structure. Based on past experience with ESIG-1000 and ESIG-500, the OM must touch approximately one cell for each potentially visible output polygon. A ratio of one OM processor for every two GP processors provides this ratio of performance.

The computational resources required for the OM to process a moving model with two fully articulated parts (primary DCS and two secondary DCSs) is equal to approximately 4 cell touches. Processing for the primary DCS only is approximately equal to 1.5 cell touches.

OM microcode currently (May 1991) uses approximately 4K words of the 8K microcode memory.

Geometric Processor

Object processing, the processing of polygons, vertices and light points, takes place in those arithmetic processors assigned as GP processors and initialized with GP microcode. Polygons and lights are transferred to the GP processors, an object at a time. While an object is being processed in each TMS320, a second object can be

read into an input queue, ready for processing. The bus controller attempts to feed a second object to each processor's input queue when the bus is idle. This it does by interrupting the processor's whose input queue is not filled with a waiting object. By double buffering objects within each processor, the input bus bandwidth can be fully utilized without having processors waste time arbiting for the bus and having to wait when the bus is busy.

Utilizing page mode access, one word per clock can be transferred from the EVM into TMS internal memory. Data can also be read from the scratch memory one word each clock. However, to transfer words from EVM to scratch memory, or to write data into scratch memory or into the output FIFO, two clocks must be utilized for each transfer. For this reason, object data, including the second buffered object, is read into and retained in internal memory. Since internal memory is 2K words long, object size has been limited to 800 words per object. This restriction is the cause of the polygons per object limitation of approximately 30.

As a polygon or light is processed, output data is written into the output FIFO memories. These memories are 4K words deep, and can buffer up an average of 100 polygons. It takes approximately 3.3 milliseconds for a GP processor to process enough data to fill the FIFO memories, if no data is being output from the FIFOs. For efficiency in the MPG, it is desirable to have the FIFOs as full as possible without becoming full before starting the MPG. Thus, the GP is started approximately 3.3 milliseconds before the MPG and DP sections of the pipe are started.

A single GP processor can process approximately 500 quadrilateral smooth shaded polygons in one 60 Hz field. When two of the three TAP processors are assigned to GP functions, a single TAP card can process 1000 polygons in a 60 Hz update period. Four-normal smooth shaded polygons take approximately 33% longer to process than the standard three-normal shaded quadrilaterals. The GP can process a three sided polygon in about 97% of the time for the standard four sided polygon.

Nondirectional lights (curved, straight, random) trade with polygons at an average ratio of 4.3 to 1. However, directional lights take approximately 2.25 times longer to process than the nondirectional lights, making them trade with polygons at the approximate ratio of 1.9 to 1.

As a GP processor encounters a change in coordinate system, it must go to EVM and read in the respective matrix and offset vector. This process is not more than about ***2%*** of the total processing effort for the entire polygon, thus, additional coordinate systems presents an insignificant load to the GP.

GP microcode is currently (May 1991) utilizing approximately 6K of microcode memory.

DISPLAY PROCESSOR

The display processor is comprised of the Clipper section, the Pixel Processor section, the Video Buffers and the optional Light Buffer for calligraphic lights.

The Clipper could be thought of as part of the GP but has been grouped with the display processor because it must operate in phase with the Pixel Processors; in other words, the Clipper and the Pixel Processors are both started at the same time with receipt of the DPGO signal, and they must both be complete before the DPDONE signal can be generated and the Bin buffer toggled. The MPB card also provides control for some sections of the pixel processors.

Clipper Section

The clipper section is made up of two cards, the Macro-poly Generator (MPG) (230114) and the Macro Poly Bin (MPB) (230113). This section of the pipeline is not expandable, but has sufficient processing capacity to accommodate a maximum system polygon load and a maximum pixel resolution.

The display screen is divided up into a uniform grid of 32-by-32 pixel areas, called macro-spans. The corners of each macro-span are defined in three-space by a set of four EP vectors, unit vectors which emanate from the eyepoint. Corner EP vectors for the four adjacent macro-spans must be identical (this is ensured within the Viewport Tool which creates the EP vectors). It is through the definition of theses EP vectors that the system can render the image using either linear or non-linear image mapping (NLIM) equations. Thus for an NLIM situation, opposite sides of a macro-span do not have to be parallel or of the same length. Since EP vectors are used to define the geometry of macro-spans whether the system is performing either linear or non-linear mapping, the ability to provide NLIM presents no additional load to the system.

In a general sense, the MPG uses the macro-span corner and edge definitions to determine which macro-spans are covered or partially covered by each polygon or light point area. A polygon filling algorithm is used which walks from macro-span to macro-span always remaining inside of the polygon. As each macro-span is determined to lie on or inside of the current polygon, the polygon data is stored in the bin associated with that macro-span. The polygon or portion of a polygon which lies inside of a macro-span is called a macro-poly. Thus, the Macro-poly Generator card generates macro-polys as it divides each polygon into multiple fragments corresponding to the macro-spans boundaries. The MPB card contains the bin memory and forms a linked list for each bin which contains all polygons found to impinge upon that macro-span.

The macro-span attribute memories store the macro-span corner vectors and the normal vectors for each of the four edges of the macro-spans. Another memory stores information relative to the boundaries of various viewports.

There is sufficient attribute memory on the MPG card and sufficient bins on the MPB card to store data for 4K macro-polys. This is four times the memory required to provide 1 million pixels. Since the system can provide multiple viewports for each channel processor, the macrospan definition memory must be divided into various regions associated with the viewports. At this level, these regions of macro-spans are referred to as viewport definitions. The regions are not called viewports because a single viewport may utilize multiple regions. Multiple viewport definitions may be defined per viewport to provide different features such as switchable fields of view, EP

vector perturbation or other effects. Figure 1 shows the 64 by 64 macro-span array divided into four viewport definition regions used to display three viewports from three Video Buffer cards.

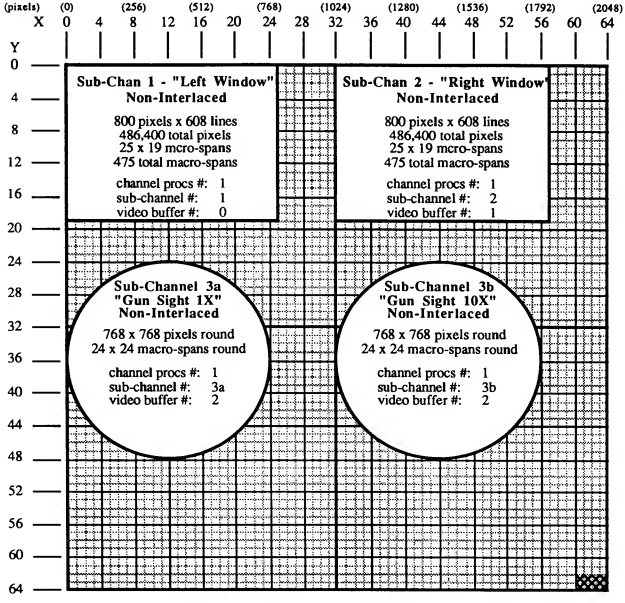


FIGURE 1 Macrospan Memory Supporting 4 Sub-channels

MPG & MPB Capacities

In a 60 Hz frame, the MPG can generate approximately 16K polygon segments called macro-polys. Also the memory on the standard MPB (230113-100) can store approximately 16K macro-polys.

There are two ways to look at the capacity of the clipper section. The first way is to examine the capability of this section of the pipe in terms of coverage factor. Unlike the display processor, the Clipper section has no knowledge of how opaque polygons fill the screen area. In other words, it knows nothing equivalent to span full in the display processor. Even though polygons are generally priority ordered, the clipper takes no advantage of keeping track of which areas of the screen have already been covered. However, for a 1K by 1K pixel area, 1000 macro-spans would be utilized, and since the clipper section can process 16K macro-polys, it has a coverage factor of sixteen--plenty for any high resolution display.

The second way to think of the clipper capacity is to consider the polygon segmentation ratio. If the system were being operated at the maximum capacity of 3,000 polygons per 60 Hz frame, then the system could handle a polygon segmentation ratio of 5.3 (16,000 divided by 3,000). There are models in the F14 program for ESIG-500 NLIM which have shown proliferation ratios of 7 or more. This is an area which needs closer study.

For configurations which process more than 3,000 polygons, such as the 15 Hz, CCTT system which may be specified at 8000 or 12,000 polygons, a bin memory limit of only 16,000 will definitely not be sufficient. The MPB card is designed to accept either the standard 1 Meg memories or optional 4 Meg memory chips. With the 4 Meg chips, the MPB can store 64,000 macro-polys, sufficient for even 12,000 input polygons. At 15 Hz operation, the MPG can also generate approximately 64,000 macro-polys.

When stuffed with the 4 Meg memory parts, the MPB card becomes the 230113-101 configuration. The ESIG-3000 will normally be configured with the 230113-100 card and ESIG-2000 is normally configured with the 230113-101 card. Thus, the higher priced system can use the lower cost card while, the lower priced system must have the high cost card.

Pixel Processors

A pixel processor is comprised of a set of three cards, the Span Assembler card (230120), the Vector Scanner card (230121) and the Texture Map card (230122). These three cards operate in concert to render the output image, one macrospan at a time.

The Pixel Processor operates on a 20 MHz clock and can generate a span's worth of information during each 50 nanosecond clock period. A span is a two-by-two pixel area, with 256 spans within a macro-span. However, since on the average, each span must be visited several times in building up the complete image, it takes longer than 256 clocks to complete a macro-span. Since there are 333,333 50-nanosecond clock periods in a 60 Hz frame, and at the rate or 4 pixels (one span) per clock, the maximum computational capacity of each pixel processor is 333,333*4*60=80 million pixels per second. The ratio of total typical pixel computational capacity divided by the number of output pixels is known as coverage factor, span touch ratio, or depth complexity. If a single pixel processor is used to compute 0.5 million output pixels at a 60 Hz frame rate, the depth complexity is computed to be about 2.65. To arrive at this

figure, various overhead tasks related to the number of polygons are also taken into account.

In the current backpanel, up to three pixel processors can be operated in parallel to provide up to 1.5 million output pixels in the 60 Hz frame time. Each pixel processor is assigned to work on a single macro-span of information at a time before it can move onto another macro-span. The pixel processors request the MPB output bus each time they need additional data. There is a small input FIFO (a few polygons deep) at the input section of each pixel processor, so that interference in arbitrating for the bus will usually be hidden in the multiple layers of buffering. Output bus bandwidth from the MPB card allows for approximately 17,000 macro-polys to be transferred in 16.67 msec.

As the macropolys in a given macrospan are rendered, the image is assembled into the span assembler ASIC chips at subpixel level resolution. The span assembler memories are double buffered, and when all elements of a macro-span have been completed, the macrospan buffers are toggled, the just-completed macro-span information is transferred to the video buffer, and at the same time, assembly of the new macro-span is begun.

Transfer of pixel information to the Video Buffers

The span assembler chip was designed to be able to output pixels at one of two rates. It can either transfer a pixels worth of information each 50 nanosecond clock period, or it can transfer the data at double that rate, one pixel transfer every 25 nanoseconds. Since four span assembler chips operate in parallel, a span can be transferred every 50 nanoseconds or every 25 nanoseconds, depending on the output mode. There are a few states of overhead associated with each macrospan transfer, so that in the slower mode, it takes 295 clocks to transfer a macrospan, and in the faster mode it takes 165 clocks.

The first version of the video buffer (230124-100) can accept pixels only at the slower rate. The Video Buffer with Sensor card (230130-100) without its daughter board can also only accept pixels at the slower rate. When the Video Buffer Expansion daughter board (230131-100) is installed, the video can be transferred at the higher rate of one span every 25 nanoseconds.

Note that at the lower transfer rate, it will take longer to transfer the macrospan to the video buffer (295 clocks) than it takes to assemble a macrospan if only one macro-poly covers the macrospan (257 clocks). If the pixel processor becomes output bus limited, the macrospan assembly process may suffer some inefficiencies. This inefficiency may be increased when more than one pixel processor must arbit for the output bus to the video buffers. For these two reasons, the video buffer input bandwidth must be considered during system configuration. In general, there should be at least 30% spare bandwidth on this bus after taking into account the resolution of the channel processor and the number of pixels which must be transferred across the bus each update period. As a guideline, any configuration above 800,000 pixels at 60 Hz update must utilize the 230130-100 video buffer with its 230131-100 daughter board.

Filtering

When the image is rendered into the span assembler buffers, the data is stored at the subpixel level, 6 samples per pixel. As the data is then transferred from the span assembler buffers to the video buffers, the six samples are averaged together to form the composite information for a single pixel. At this filtering level, no information from adjacent pixels is considered. Thus, as the pixels enter the video buffers, they have been filtered with a 1-pixel-by-1-pixel flat filter.

As the pixels are read from the video buffer for displays, additional image filtering may take place. Two lines of video information are read from the video buffers concurrently and can be averaged together. Adjacent pixels along the line can also be averaged during this post-filtering process. Three options are available for image filtering. First, the data can be displayed directly as it was stored in the video buffers, prefiltered with a 1-by-1 flat filter. Second, a two-by-two pixel flat post filter can be applied. Third, a two-high-by-one-wide pixel flat post filter can be applied. The user can specify the desired image filter mode with a command in the site file or interactively at the system console using the "FILTER" command.

It is envisioned that for interlaced operation, either the 2-by-1 or 2-by-2 filter should be used to minimize temporal aliasing effects.

It is important to note here that with the two-stage filter process, the system can support only flat-shapped antialiasing filters. Since macro-spans do not overlap, and since information relative to pixels in adjacent macro-spans is not known, the first filter stage cannot consider information from adjacent pixels. In order to support a non-flat filter, such as a triangular or Gaussian filter, subpixel information from adjacent pixels would need to be taken into consideration in a single filtering operation; this implies a more complicated and costly architecture.

Span Assembler Card (230120)

The span assembler card reads macro-poly edge data from the Macro Poly Bin card across the MPB bus. A single level of buffering external to the Picker ASIC receives the macro-poly data before processing. A macro-poly can be read from the MPB in 19 clock cycles. The Picker chip then determines which spans are covered or partially covered by the macro-poly. This it does in a hierarchical, tightly packed pipeline process which can accommodate several polygons in the pipeline at one time. At the output of the Picker chip, span addresses are output once each clock period. If successive macro-polys cover less then 19 spans each (there are 256 spans per macrospan), then the output rate can exceed the input rate of macro-polys from the MPB and the pipeline cannot operate at maximum efficiency. When the input cannot keep up with the output, then invalid spans are output with the invalid flag set indicating to future processors that the span output should be ignored. Since multiple macro-polys can be buffered in the Picker chip at one time, a mixture of larger and smaller macro-polys will prevent output inefficiencies. Only when several small macro-polys are encountered in succession will the pipeline operate inefficiently.

Span addresses are sent to the Vector Scanner and Texture Map cards so that the proper texture, range and shading values can be computed for each span. The Vector Scanner and Texture Map algorithms are computed in a strictly pipeline fashion, generating their span output data each clock cycle, and can thus never hold up or delay the rendering of spans.

Range, transparency and color data are returned from the TM and VS cards to be assembled in the Span Assembler chips on the Span Assembler card.

One, Two and Four Sample Mode Operation

In the normal mode, where one span is processed each clock period, a single texture sample and range sample are computed and applied across the entire span (four pixels). In this mode, texture resolution is only a quarter as good as in the ESIG-1000 and a little less than half as good as in the ESIG-500/600HT. (In the ESIG-500, texture is sampled every other pixel along the line, and the intermediate pixel texture value in an average of the two adjacent values.) Also in this mode, a single range value is computed for the span, and this range value is stored in the R-buffer at the sample level, six samples per pixel. Since the range is stored and compared independently for each of the six samples, antialiasing of polygon edges is preserved, even those which are R-buffered. However interpenetrating polygons will exhibit span sized jaggies.

There are two optional modes of operation, a two-sample mode and a four-sample mode. The four-sample mode will be discussed first. In this mode, the assembly of data for a span is slowed by a factor of four. During four clock cycles, four texture and four range values are computed, one for each of the pixels in the span. Thus in this mode, texture resolution increases by a factor of four. Also for interpenetrating polygons, the jaggies go down to pixel-sized jaggies, a much more acceptable level. However, the penalty for this extra texture and R-buffer resolution, is a reduction of four in the rendering speed. If the entire picture were rendered in this mode, the coverage factor would have to be greater than 8, to compensate for the increase by a factor of four in rendering time. The saving principle here is that four-sample mode rendering can be enabled on an individual polygon basis. Thus, if a few small polygons require higher texture or R-buffer sampling, they can employ this mode without an intolerable load increase to the system. The two- or four-sample mode can also be enabled on a macro-span basis, providing for a "poor man's" higher resolution inset. There are also future plans to monitor the DP processing load and disable four-sample mode on overload conditions.

In the two-sample mode, which can also be enabled on an individual polygon bases, two texture and Range sample are made per span with a penalty of doubling the processing time. These samples are displaced in the horizontal direction. When two-sample mode is used in conjunction with EP vector perturbation, an effective texture and range sampling of four samples per span can be achieved. However, perturbation is only useful for interlaced, field rate update systems.

Vector Scanner Card

The Vector Scanner card creates the two independent texture scanning addresses (u, v pairs), given the macrospan addresses from the Span Assembler card. These span level u, v pairs are sent to the Texture Map card and resulting texture values are returned. The Vector Scanner card also creates the range to the polygon, the resultant fog and smooth shading values, range attenuation information, and the landing lobe value for the span. Texture, fog, landing lobe, color, smooth shading and polygon intensity information are then combined within the Hue chips to provide the resultant polygon color values to be applied to the span. Range and RGB color values are then sent to the Span Assembler card to be stored in the range buffer and the span buffer where appropriate.

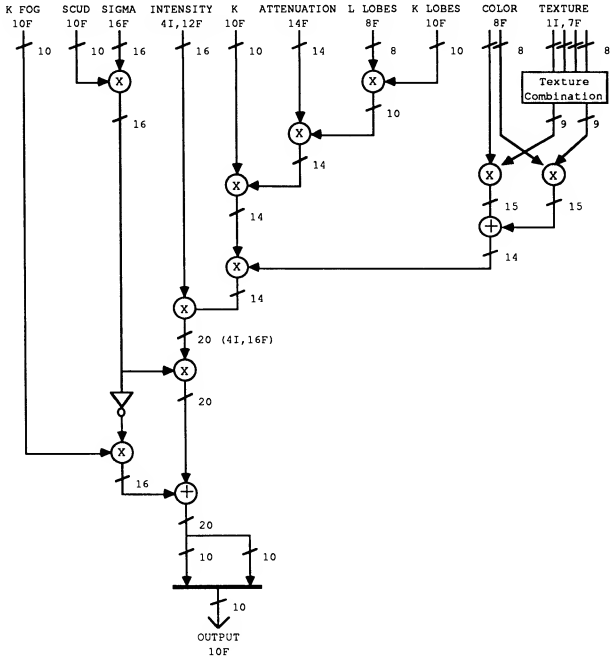


FIGURE 2. HUE CHIP DATA PATH WIDTHS

Figure 2 shows the basic data path widths inside of the Hue chip. In the notation, I stands for integer bits and F for fractional bits. This section of the system has been specifically designed to provide maximum intensity resolution. There are three Hue chips on the Vector Scanner card, one for red, one for green and one for blue. In normal RGB mode, each color component is output with ten bits of resolution. ESIG-500 performed most intensity calculations at 8 bits, only retaining 10 bits from the final multiplier. As a result, ESIG-500 demonstrated intensity Mach bands on dark polygons, during night/dusk situations, and in NVG operation. Ten bits of resolution in

ESIG-3000 in the Sigma path will provide superior performance to most systems which utilize only 8 bits per color component.

Enhanced Resolution in NVG and FLIR Modes

The ESIG-3000 system has incorporated extra high computational resolution in the intensity calculations to enhance its use in FLIR and NVG modes of operation. As can be seen from Figure 2, polygon intensity can be specified with 3 integer and 12 fractional bits of resolution, giving a wide dynamic contrast range. The incoming intensity bits, including any smooth shading calculations, are also computed with 15 bits of resolution. Sigma (used for fog attenuation) is computed with 16 bits, so that full resolution is maintained through with fog attenuation. After the intensity and sigma multiply, which is the final operation, 20 bits of data result. The result is a signed positive value, so only 19 bits of resolution is realizable.

In FLIR mode, the red intensity is discarded. Under special control, identical computations take place in the blue and green Hue chip, each resulting in 20 bits of resolution. At the final data multiplexing stage, the green chip outputs the upper 10 bits of result, and the blue chip outputs the lower 10 bits of result. This provides the Span Assembler card with a 20-bit monochrome intensity value. This 20-bit value is eventually stored on the Video Buffer card in the combined green and blue data fields. The Video Buffer with Sensor card (230130) uses all 20 bits in the Filter chip to provide a large range of level and gain control. Gains as high as plus or minus 64 can be accommodated while still maintaining ten bits of good output data to the DAC.

Texture Map Card (230122)

The Texture Map card contains the texture memories, the addressing circuitry into those texture maps, the bilinear blend circuitry following the maps and other functions associated with texture such as clamping, contrast management and various other texture combinatorial circuitry.

The texture data (four bits per texel) is stored in static 4-bit-wide memory chips. The standard configuration of Texture Map card (230122-100) uses 64K x 4 memory chips, providing a total of 1 Meg of texels or 64 128x128 maps. With the expanded Texture Map card option (230122-101) 256K x 4 memories are used, providing 4 Meg of texels or 256 128x128 maps.

Immediately after the four-bit texture values are read from texture memory, the four bits are expanded into eight bits and all subsequent computations are performed at the eight bit level. The mapping of four bits into eight bits for texels which represent intensity or color modulation is rather straight forward. A 1 becomes a 11, a 2 becomes a 22, a 3 becomes a 33 and so forth to an F becomes an FF (all in hexadecimal, of course). The mapping of four bit distance data for contour texture is not so straight forward. Distance data is logarithmic in nature. The mapping of transparency is such that the 16 possible texture values map into the 25 levels of transparency in a linear manner, with the endpoint texture values representing fully opaque and fully transparent.

There are four banks of texture memory, independently addressed to provide a combination of up to four texture maps per polygon, each with a different scale. Each of the four banks holds one quarter of the total texture maps. The memory in each bank is arranged in four sub-banks to provide look-up of four adjacent texels each clock. The four adjacent texels are bilinear blended to provide a single texture value from that bank for each span.

The four banks of texture are further grouped into two groups of two banks each. Each of the two major groups, group A and group B, are addressed using completely independent U,V coordinates from each polygon and independent scanning hardware. The first bank in each group is called the independent bank, and the second bank is known as the dependent bank. Thus, we have four banks, A independent (AI), A dependent (AD), B independent (BI), and B dependent (BD). Addressing for a dependent bank is a scaling of the addressing to its respective independent bank from the same group.

The Texture Map card and related system hardware and software support many different types of texture. Contour texture maps are supported and can reside in texture bank AI only. Full color texture requires three of the four maps, one each, for red, green and blue. The remaining map can be used to produce contour, transparency or an additional modulation on the polygon. A two color texture is supported where the A texture bank can modulate one color (a complete RGB triplet) and the B texture bank can modulate a second color (RGB triplet) before the two texture results are added together. This should result in a significantly more powerful color modulation capability then the traditional color blend texture (which is also supported). A complete description of the texture combinations appears in the Chapter entitled P-100 Texture Overview.

The Texture Map card outputs to the Vector Scanner card four eight-bit texture values each clock period. The combination of these four texture values takes place within the Hue chips on the Vector Scanner card.

All transparency data, whether as a result of texture or from polygon data, goes directly to the Span Assembler card.

Video Buffer Section

Five slots in the backpanel are reserved for the Video Buffer section of the IG. Up to four Video Buffer cards (230124 or 230130) may be installed. Note that 230124 and 230130 cards cannot be mixed in the same channel processor. The fifth slot is reserved for the Light Buffer card (230123) for those systems requiring calligraphic lights.

Each macrospan, as it is being output from the MPB card, has associated with it the video buffer address, the specific memory address within the video buffer, and a subchannel code. This mapping of bin number to video buffer address is made via a table generated by the Viewport Tool and stored on the MPB card. Using this technique, data associated with a given viewport can be stored into, and output from, the appropriate VB card. A viewport may also be mapped to more than one VB card

for unique users, such as for the SMTF program, who want the same viewport data output from two VB cards, each modified by different gamma functions.

Video Buffer (230124-100) card

Each Video Buffer (VB) card provides a double-buffered, full frame store for the rendered image. The output buffer can output the video in either an interlaced or non-interlaced format. Normally, the video will be refreshed from the output buffer at 50 or 60 hertz. Since a full frame store is employed, both interlaced fields can be correctly output even when the buffers are toggled at a rate slower than the refresh rate.

The two sides of the frame buffer are implemented using standard 256K by 4 bit DRAM memories, addressed in the static column mode. Ten bits each, of red, green and blue video data is stored within a 1024 by 1024 array of pixel data. Two lines of video data can be read out simultaneously to allow for the post filter to cover an area two lines high.

The control for transferring data from the Span Assembler card(s) to the various Video Buffer card(s) resides on the MPB card. When a Span Assembler card completes the rendering process for a macrospan, it requests the output bus, the controller on the MPB grants access to the bus and controls transfer of the data to the appropriate VB card. This same controller provides memory control signals (RAS, CAS and refresh) for the input buffer. Each double row of pixels (16 spans) in a macrospan is written into the buffer in the static column mode, with a new column access between each span row.

The output side of the buffer receives its control from the CEM card. This control includes memory control signals (RAS, CAS, read and refresh) as well as all video sync and timing signals. Since this single control on the CEM card controls all VB cards in parallel, they must all operate in synchronism, with the same scan standards and resolution. The memory is organized such that a static column contains 1024 adjacent locations, and must correspond to a single line of video. The video line need not contain 1024 pixels, but the entire video line must be fully contained in a single static column since a row read cannot take place during the output of a line of video.

Gamma correction for the output video is provided on the VB card. Separate gamma correction tables are available for red, green and blue video. In the 230124 VB up to 16 separate gamma functions can be stored on line simultaneously. Selection of the gamma function takes place at the viewport level. Thus, each viewport, including a multiple viewport presented on a shared display, can have a different gamma function applied to it. Only 4 on-line gamma functions will be available in the 230130 VBS card.

On each VB card there are three connectors providing parallel output display drive signals. These three output ports may be used to drive three separate displays in parallel. Where high impedance inputs are provided in the display device, a single video output cable can drive multiple display devices by daisy-chaining the video cable to each display and terminating the cable at the last display device.

The video controller on the CEM card can be programmed to provide separate horizontal and vertical sync signals, composite sync, or sync on green. The 230124-100 card is limited in maximum output video rate to a pixel every 25 nanoseconds. This allows interlaced output video up to the full 1024 x 1024 resolution. Non-interlace operation is only possible up to approximately 600,000 pixels. The Video Buffer output specification provides more detailed information.

X. RELATED PROCEEDINGS APPENDIX

(No matter presented)